

# Iterative Detection for Compressive Sensing: Turbo CS

Amin Movahed, Mark C. Reed

School of Engineering and Information Technology, University of New South Wales, Canberra, Australia  
a.movahed@student.unsw.edu.au, mark.reed@unsw.edu.au

**Abstract**—We consider compressive sensing as a source coding method for signal transmission. We concatenate a convolutional coding system with 1-bit compressive sensing to obtain a serial concatenated system model for sparse signal transmission over an AWGN channel. The proposed source/channel decoder, which we refer to as turbo CS, is robust against channel noise and its signal reconstruction performance at the receiver increases considerably through iterations. We show 12 dB improvement with six turbo CS iterations compared to a non-iterative concatenated source/channel decoder.

## I. INTRODUCTION

In real transmission systems, source coding is used to minimize the transmitted bits. Moreover, channel coding is nearly always applied to minimize bit errors due to the channel noise. Therefore, source coding concatenated with channel coding is a recognized approach for reliable transmission of data [1].

Compressive sensing (CS) is a new source coding approach in which signal measurement and compression are performed in a single step. The basic idea of CS is that any  $N$ -dimensional signal which is  $K$ -sparse (i.e., there are only  $K$  non-zero elements in the signal where  $N \gg K$ ) is measured through few random linear projections. The sufficient number of projections,  $M$ , guaranteeing signal reconstruction is often much less than  $N$  [2]. Thus, CS can be considered as a method of data compression with rate  $N/M$ . However, CS deals only with sparse or approximately sparse signals [3]. In practice, many types of signals are sparse or can be represented with a sparse vector in a proper basis. Moreover, in some signal processing applications, e.g., magnetic resonance imaging, the processes of measurement and compression are not separable, and acquiring the signal through linear projections is an intrinsic part of the measuring process [4].

In this paper, we use the principle of concatenated codes and turbo coding. Turbo codes are powerful channel encoding techniques first introduced by Berrou *et al.* in 1993 [5] and the decoding performance achieves results close to the channel capacity. The encoding structure of a turbo encoder consists of a serial or parallel concatenation of convolutional encoders separated by random interleaving.

In particular, we utilize the serial concatenated code approach [6]. The serial turbo decoder signal is decoded in an iterative process between two *a posteriori probability* (APP) soft-input/soft-output decoders [7].

The aim of this work is to apply a source encoder as the outer encoder concatenated with an inner channel decoder.

In [8], the authors introduced a turbo decoding approach by concatenating fixed length codes with convolutional codes for audio/video transmission. In this paper, we apply CS as a generic source coder for any kind of sparse signal. In order to do so, there are two main challenges:

- to input *a posteriori* belief provided by the APP decoder to the CS decoder.
- to calculate *a priori* information from the CS decoder as input to the APP decoder for the next iteration.

As an approach, Bayesian CS [9], which is a CS decoding method considering CS inversion from a Bayesian prospective, could be applied. Bayesian CS provides density function for each element of the reconstructed signal, which can be applied as *a priori* information. However, the output of CS encoders is zero mean Gaussian distributed values while the input of convolutional encoders are  $-1$  and  $+1$ . Thus, a special quantization is needed after a CS encoder.

In this work, we use 1-bit CS as the outer encoder. 1-bit CS is a quantized version of CS representing each measurement by only a two-state value [10].

There are several methods introduced in the literature to solve 1-bit CS decoding problem. Some of these methods are based on linear and convex programming, e.g. [11], [12], while others are based on greedy methods [10], [13]–[18]. However, all the above mentioned methods only accept binary values as input to estimate the signal. In addition, none of these methods generates soft-valued *a priori* information.

The key contribution of this paper is to propose a new reconstruction method for 1-bit CS which accepts soft-input and generates soft-output and, hence, is able to work iteratively together with an APP decoder to reconstruct the signal at receiver in the same fashion as in a classic serial concatenated turbo code.

We refer to the proposed coding approach as turbo CS coding. The turbo CS encoder consists of the concatenation of a 1-bit CS encoder and a convolutional encoder at the transmitter. In the receiver, the turbo CS decoder iterates between an APP decoder and a 1-bit CS decoder. Numerical experiments show a significant improvement in the quality of the reconstructed signal through turbo CS iterations.

## II. SYSTEM MODEL

In this section, we describe the serial concatenated transmission and channel model. In the first part, we discuss 1-bit

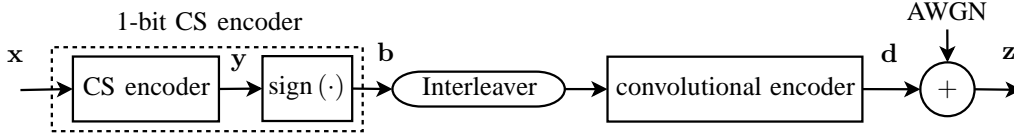


Fig. 1: Transmitter model

CS configuration and in the second part we combine 1-bit CS with a convolutional encoder.

#### A. 1-bit compressive sensing

In classic compressive sensing, each measurement  $y$  is obtained through a projection of  $K$ -sparse signal,  $\mathbf{x} \in \mathbb{R}^N$ , onto a random vector  $\phi \in \mathbb{R}^N$ . Therefore, for  $M$  number of measurements ( $M < N$ ) we have

$$\mathbf{y} = \Phi \mathbf{x} \quad (1)$$

where  $\phi_i$  is the  $i$ th row of  $\Phi \in \mathbb{R}^{M \times N}$  and  $\mathbf{y} = [y_1, y_2, \dots, y_M]^T$ . It is shown that exact signal reconstruction is guaranteed when  $\Phi$  satisfies the restricted isometry property [3].

In most practical cases, obtained measurements need to be quantized before reconstruction. In the extreme case, which is referred to as 1-bit CS, measurements are represented by only one bit [10]. 1-bit CS output is essentially a sign function over CS measurements. Hence, binary measurements,  $\mathbf{b} \in \{-1, 1\}^M$ , are obtained from

$$\mathbf{b} = \text{sign}(\mathbf{y}) = \text{sign}(\Phi \mathbf{x}) \quad (2)$$

where  $\text{sign}(\cdot)$  denotes the sign function.

#### B. Serially concatenated encoders

At the transmitter, the interleaved binary output of the 1-bit CS encoder is encoded by a convolutional encoder. We denote the coded bits by  $\mathbf{d} \in \{-1, +1\}^P$ .  $M/P$  is the rate of the convolutional encoder. In turbo coding context, the 1-bit CS encoder and the convolutional encoder are referred to as outer and inner encoders respectively. The coded bits are transmitted through an AWGN channel with a known variance,  $\sigma_n^2$ . The channel output is then

$$\mathbf{z} = \mathbf{d} + \mathbf{n} \quad (3)$$

where  $[\mathbf{n}]_i \sim \mathcal{N}(0, \sigma_n^2)$  and  $[\cdot]_i$  denotes the  $i$ th element in the argument. The system model is illustrated in Fig. 1.

In the next section, we propose an iterative method to reconstruct  $\mathbf{x}$  at the receiver from the noisy coded measurements  $\mathbf{z}$ .

### III. ITERATIVE 1-BIT COMPRESSIVE SENSING: TURBO CS

#### A. A posteriori probability decoder

A *a posteriori probability* (APP) decoder is a soft-input/soft-output decoder [7]. APP takes two inputs: received signal  $\mathbf{z}$  and *a priori* probability of elements of  $\mathbf{b}$  denoted by  $\alpha$ . Hence, we have

$$\mathbb{P}([\mathbf{b}]_i = +1) = [\alpha]_i \quad \text{and} \quad i = 1, \dots, M. \quad (4)$$

At the output, APP gives *a posteriori* probability of the elements of  $\mathbf{b}$  denoted by  $\alpha'$ . Therefore,

$$\mathbb{P}([\mathbf{b}]_i = +1|\mathbf{z}) = [\alpha']_i \quad \text{and} \quad i = 1, \dots, M. \quad (5)$$

Typically in a *maximum a posteriori probability* decoder, a decision is made on  $\alpha'$  yielding hard-bits.

In iterative decoders, however, bit probabilities are exchanged between decoders, since they contain information about the reliability of the data. A vector containing soft-bits is denoted by  $\mathbf{b}_{\text{soft}} \in \{[-1, 1]\}^M$ . Each element in  $\mathbf{b}_{\text{soft}}$  is defined as the expected value of the corresponding element in  $\mathbf{b}$ . Hence, for *a priori* soft-bits we have

$$\begin{aligned} [\mathbf{b}_{\text{soft}}]_i &= \mathbb{E}([\mathbf{b}]_i) = \mathbb{P}([\mathbf{b}]_i = +1) - \mathbb{P}([\mathbf{b}]_i = -1) \\ &= [\alpha - (1 - \alpha)]_i = 2[\alpha]_i - 1. \end{aligned} \quad (6)$$

In the same way, *a posteriori* soft-bits are obtained from

$$[\mathbf{b}'_{\text{soft}}]_i = \mathbb{E}([\mathbf{b}]_i | \mathbf{z}) = 2[\alpha']_i - 1. \quad (7)$$

Furthermore, hard-bits are denoted with  $\mathbf{b}_{\text{hard}}$  and we have

$$\mathbf{b}_{\text{hard}} = \text{sign}(\mathbf{b}'_{\text{soft}}). \quad (8)$$

Intuitively, when  $\mathbb{P}([\mathbf{b}]_i = +1)$  is 0, the  $i$ th soft-bit is  $-1$  and when  $\mathbb{P}([\mathbf{b}]_i = +1)$  is 1, the  $i$ th soft-bit is  $+1$ .

In iterative decoding, the inner decoder needs to receive the parameter  $\mathbf{b}_{\text{soft}}$  and estimate  $\mathbf{x}$  and  $\mathbf{b}'_{\text{soft}}$ . In the next two sections, we give a brief review on 1-bit CS reconstruction and then introduce a 1-bit CS algorithm that can be used in an iterative turbo CS decoder where the CS constituent decoder accepts soft bits in and generates soft bits out.

#### B. 1-bit CS reconstruction algorithm

The aim of a 1-bit CS reconstruction algorithm is to estimate the values in a vector  $\mathbf{x}$  based on an observation vector  $\mathbf{b}$  and knowing the measuring matrix  $\Phi$ . In many practical cases, there might be some random bit flips in  $\mathbf{b}$  due to the quantization error or noise in the transmission process. The number of these bit flips is a measure of the noise level. Some of the reconstruction algorithms consider the number of the bit flips to reconstruct the signal efficiently and are robust against the random bit flips in the binary measurements [17], [18].

Among all 1-bit CS reconstruction algorithms, *adaptive outlier pursuit with bit flips* (AOP-f) [17] has the best reconstruction performance in the presence of random bit flips and when the sparsity level of the signal and the number of the bit flips are known. There are two types of AOP-f based on  $\ell_1$ -norm minimization (AOP- $\ell_1$ -f) and  $\ell_2$ -norm minimization (AOP- $\ell_2$ -f). Since AOP- $\ell_1$ -f outperforms AOP- $\ell_2$ -f in terms of

signal reconstruction performance, we focus on AOP- $\ell_1$ -f in this paper. Henceforth, we refer to AOP- $\ell_1$ -f as AOP-f.

AOP-f is an iterative algorithm that estimates  $\mathbf{x}$  and the position of the bit flips in  $\mathbf{b}$ .  $\tilde{\mathbf{b}}$  denotes the noisy binary measurements vector and  $L$  denotes the number of the bit flips in  $\tilde{\mathbf{b}}$ . The position of the random bit flips in  $\tilde{\mathbf{b}}$  is represented by vector  $\boldsymbol{\Omega} \in \{-1, 1\}^M$  where  $\boldsymbol{\Omega} = \mathbf{b} \odot \tilde{\mathbf{b}}$  and  $\odot$  denotes element-wise product. That is,  $[\boldsymbol{\Omega}]_i = -1$  means that there is a bit flip in  $[\tilde{\mathbf{b}}]_i$ . AOP-f solves the following optimization problem

$$\begin{aligned} (\hat{\mathbf{x}}, \hat{\boldsymbol{\Omega}}) = \arg \min_{\mathbf{x}, \boldsymbol{\Omega}} & \left\| (\tilde{\mathbf{b}} \odot \boldsymbol{\Omega} \odot \Phi \mathbf{x})^- \right\|_1 \\ \text{s.t. } & \frac{1}{2} \sum_i (1 - [\boldsymbol{\Omega}]_i) \leq L \\ & \|\mathbf{x}\|_0 \leq K \\ & \|\mathbf{x}\|_2 = 1 \end{aligned} \quad (9)$$

where  $\|\cdot\|_p$  denotes  $\ell_p$ -norm<sup>1</sup> of the argument and  $(\cdot)^-$  is negative function defined as

$$([\mathbf{x}]_i)^- = \begin{cases} |[\mathbf{x}]_i|, & \text{if } [\mathbf{x}]_i < 0, \\ 0, & \text{otherwise.} \end{cases}$$

In the next section, we propose some changes to the input of AOP-f to be able to utilize soft-bits as input. In addition, we apply a mapping method on the reconstructed signal to produce *a priori* soft-bits to be used as an input to the APP decoder.

### C. Soft-in/soft-out 1-bit CS decoder

As mentioned in section III-B, AOP-f accepts binary values as input to reconstruct the signal. Therefore, a trivial way to apply AOP-f as a decoder after the APP decoder is to use  $\mathbf{b}_{\text{hard}}$  from (8) in (9). However, by solely using hard-bits, we lose information about the reliability of the data. In addition, AOP-f needs to know an estimate of the number of the bit flips in  $\mathbf{b}_{\text{hard}}$  to reconstruct the signal efficiently.

Here, we develop a method to use soft-bits as input to reconstruct the signal via AOP-f.  $\tilde{\mathbf{b}}$  is replaced with  $\mathbf{b}_{\text{hard}}$  in (9). In addition, we define  $\boldsymbol{\alpha}_{\text{flip}}$  whose elements represent the probability of a bit flip in the corresponding element of  $\mathbf{b}_{\text{hard}}$ . Thus,  $\boldsymbol{\alpha}_{\text{flip}}$  is derived from

$$[\boldsymbol{\alpha}_{\text{flip}}]_i = \begin{cases} \mathbb{P}([\mathbf{b}]_i = -1 | \mathbf{z}), & \text{if } [\mathbf{b}_{\text{hard}}]_i = 1, \\ \mathbb{P}([\mathbf{b}]_i = +1 | \mathbf{z}), & [\mathbf{b}_{\text{hard}}]_i = -1. \end{cases} \quad (10)$$

Substituting (5), (7) and (8) in (10) gives

$$[\boldsymbol{\alpha}_{\text{flip}}]_i = \begin{cases} 1 - [\boldsymbol{\alpha}']_i, & \text{if } [\boldsymbol{\alpha}']_i \geq 0.5, \\ [\boldsymbol{\alpha}']_i, & \text{otherwise.} \end{cases} \quad (11)$$

The estimated number of the bit flips is denoted by  $\bar{L}$  and is obtained from

$$\bar{L} = \text{round} \left( \sum_{i=1}^M [\boldsymbol{\alpha}_{\text{flip}}]_i \right). \quad (12)$$

<sup>1</sup>  $\|\mathbf{x}\|_p := \left( \sum_{i=1}^N |[\mathbf{x}]_i|^p \right)^{1/p}$

Now with  $\mathbf{b}'_{\text{soft}}$  from (7) and  $\bar{L}$  from (12),  $\mathbf{x}$  can be estimated through AOP-f and the following optimization can be solved via the algorithm in [17]

$$\begin{aligned} (\hat{\mathbf{x}}, \hat{\boldsymbol{\Omega}}) = \arg \min_{\mathbf{x}, \boldsymbol{\Omega}} & \left\| (\text{sign}(\mathbf{b}'_{\text{soft}}) \odot \boldsymbol{\Omega} \odot \Phi \mathbf{x})^- \right\|_1 \\ \text{s.t. } & \frac{1}{2} \sum_i (1 - [\boldsymbol{\Omega}]_i) \leq \bar{L} \\ & \|\mathbf{x}\|_0 \leq K \\ & \|\mathbf{x}\|_2 = 1. \end{aligned} \quad (13)$$

The next step of the decoder generates soft-bits,  $\mathbf{b}_{\text{soft}}$ , at the output. We apply a CS encoder over the estimated signal. Thus, we obtain

$$\mathbf{y}' = \Phi \hat{\mathbf{x}}. \quad (14)$$

Elements of  $\mathbf{y}'$  can be approximated by a Gaussian distribution with zero mean. In this case, unlike *binary phase shift keying* (BPSK) system, most of the received values to be mapped are concentrated around 0. The challenge is to map these values to an interval between  $-1$  and  $1$  based on their reliabilities. The elements with values around 0 are the least reliable for generating *a priori* soft-values. The elements with the most reliability are the ones that are the furthest from 0. Therefore, we utilize elements of  $\mathbf{y}'$  that are further from 0, and over iterations, we consider the influence of the elements of  $\mathbf{y}'$  with values closer and closer to zero.

In the case that either there is no noise in the received binary measurements or the estimation of the number of the bit flips is exact,  $\mathbf{y}'$  is very close to  $\mathbf{y}$  and the sign of each element of  $\mathbf{y}'$  describes the sign of the corresponding element in  $\mathbf{b}$ . In the noisy case, however, there are some sign mismatches between the elements of  $\mathbf{y}'$  and  $\mathbf{y}$ . To consider the effect of the random bit flips on the soft-values, we multiply  $\mathbf{b}_{\text{hard}}$  with  $\mathbf{y}'$  and the result is denoted by  $\boldsymbol{\psi}$ ,

$$\boldsymbol{\psi} = \text{sign}(\mathbf{b}'_{\text{soft}}) \odot \mathbf{y}' = \mathbf{b}_{\text{hard}} \odot \mathbf{y}'. \quad (15)$$

In fact, the element-wise multiplication in (15) removes the sign of the elements of  $\mathbf{y}'$ . In the case that there is no bit flips in  $\mathbf{b}_{\text{hard}}$ , then  $\mathbf{b}_{\text{hard}} = \mathbf{b} = \text{sign}(\mathbf{y}')$  and all the elements of (15) are positive. However, in the presence of the random bit flips, the negative elements of  $\boldsymbol{\psi}$  depict the sign flips in  $\mathbf{b}_{\text{hard}}$  and the elements with large amplitudes are more reliable than the ones with small and negative amplitudes. Based on the above facts, a mapping function is introduced which maps each element of  $\boldsymbol{\psi}$  to a real value between  $-1$  and  $1$ . The mapping function  $\Lambda(\boldsymbol{\psi})$  is defined as follows

$$\Lambda(\boldsymbol{\psi}) = \begin{cases} \min \left( 1, \frac{[\boldsymbol{\psi}]_i}{\gamma \cdot \max(\boldsymbol{\psi})} \right), & \text{if } [\boldsymbol{\psi}]_i \geq 0, \\ \frac{[\boldsymbol{\psi}]_i}{\lceil \min(\boldsymbol{\psi}) \rceil}, & [\boldsymbol{\psi}]_i < 0 \end{cases} \quad (16)$$

where  $0 \leq \gamma \leq 1$  is the normalized Euclidean distance between  $\mathbf{b}'_{\text{soft}}$  and  $\mathbf{b}_{\text{hard}}$ . We have

$$\gamma = \frac{\|\mathbf{b}'_{\text{soft}} - \text{sign}(\mathbf{b}'_{\text{soft}})\|_2}{\sqrt{M}} = \frac{\|\mathbf{b}'_{\text{soft}} - \mathbf{b}_{\text{hard}}\|_2}{\sqrt{M}}. \quad (17)$$

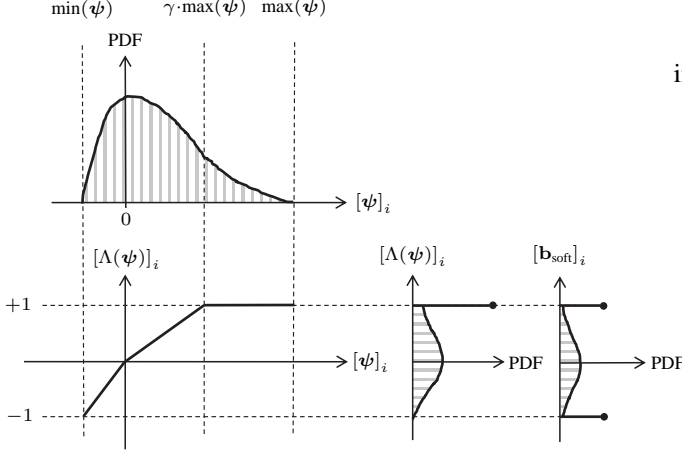


Fig. 2: Mapping method

In fact,  $\gamma$  determines how much information is lost by applying sign function over  $\mathbf{b}'_{\text{soft}}$ .

Since the signs of the elements in  $\mathbf{y}'$  were removed in (15), the obtained values from (16) need to be multiplied again by  $\mathbf{b}_{\text{hard}}$  in order to bring the signs back. Hence, the soft-output is obtained by

$$\mathbf{b}_{\text{soft}} = \Lambda(\psi) \odot \mathbf{b}_{\text{hard}}. \quad (18)$$

In Fig. 2, the mapping method is depicted. In words,  $\Lambda(\psi)$  is a mapping function that categorizes the elements of  $\psi$  by their signs:

- The negative elements of  $\psi$  are mapped to values in an interval between  $-1$  and  $0$  based on their amplitudes. As mentioned above, the negative elements in  $\psi$  specify the bit flips in  $\mathbf{b}_{\text{hard}}$ . In addition, the negative elements with small values are more likely to be flipped and are mapped to values close to  $-1$ .
- The positive elements of  $\psi$  are mapped based on their amplitudes between  $0$  and  $\gamma \cdot \max(\psi)$  to values between  $0$  and  $+1$ . Elements of  $\psi$  exceeding  $\gamma \cdot \max(\psi)$  are clipped and mapped to  $+1$ .

We refer to the proposed decoding method as soft-in/soft-out 1-bit CS decoder.

*Example:* To justify the performance of the soft-in/soft-out 1-bit CS decoder, we consider the best case where there is no noise in the binary measurements. Hence,  $[\alpha_{\text{bit}}]_i = 0$  for  $i = 1 \dots M$ . We have  $\bar{L} = 0$  from (12).  $\hat{\mathbf{x}}$  is estimated by (13). Elements of  $\psi$  obtained from (15) are all positive values. Therefore,  $\min(\psi) = 0$ . Furthermore,  $\text{sign}(\mathbf{b}'_{\text{soft}}) = \mathbf{b}_{\text{hard}}$  and (17) gives  $\gamma = 0$  that yields  $\gamma \cdot \max(\psi) = 0$ . Thus, all the elements of  $\Lambda(\psi)$  are 1. In this case,  $\mathbf{b}_{\text{soft}}$ , given by (18), is identical to  $\mathbf{b}'_{\text{soft}}$ .

#### D. Combination of soft-in/soft-out 1-bit CS and APP decoding

In section III-C, the soft-in/soft-out 1-bit CS reconstruction method was introduced which receives soft-bits and generates improved soft-bits as output. In this section, we combine the soft-in/soft-out 1-bit CS decoder with an APP decoder to

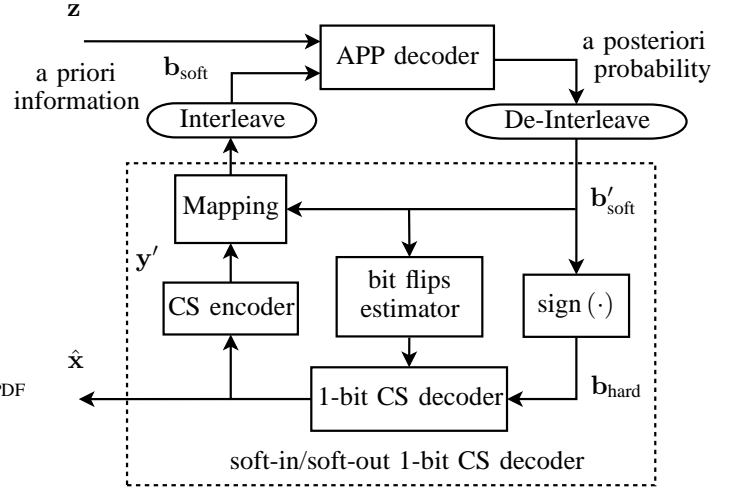


Fig. 3: Turbo CS decoder

obtain the turbo CS decoder for the transmission system in section II.

As discussed in section II, the transmission system consists of a 1-bit CS encoder serially concatenated with a convolutional encoder at the transmitter. Hence, the 1-bit CS encoder works as a source encoder that receives real values and compresses the data with rate  $M/N$ . The binary output of the 1-bit CS encoder is given to the convolutional encoder. At the receiver, as illustrated in Fig. 3, the received noisy signal is input to an APP decoder. The *a priori* soft-bits are zero for the first iteration. The soft-output of the decoder, namely *a posteriori* probability, is given to the soft-in/soft-out 1-bit CS decoder to estimate the transmitted signal. The soft-output of the soft-in/soft-out 1-bit CS decoder is provided to the APP decoder as *a priori* information for the next iteration. These steps are repeated for each iteration. Through the iterations and as  $\mathbf{b}'_{\text{soft}}$  tends to  $\mathbf{b}$ ,  $\gamma$  goes to 0 and the output of the turbo CS decoder converges.

## IV. NUMERICAL RESULTS

In this section, we verify the reconstruction performance of turbo CS through numerical simulation. We choose  $K$ -sparse signal vector  $\mathbf{x}$  randomly in each realization. We set the dimension of the signal  $N = 1000$  and its sparsity level  $K = 10$ . The non-zero elements of  $\mathbf{x}$  follow zero-mean Gaussian distribution with variance 1. These elements are distributed uniformly through the signal vector  $\mathbf{x}$ . The elements of measuring matrix  $\Phi$  are generated based on a Gaussian distribution with zero mean and variance  $1/M$ . The number of the encoded bits is set to  $M = 500$ . Thus, the rate of the 1-bit CS encoder is  $N/M = 2$ . The signal is encoded through the 1-bit CS encoder and its binary output is interleaved by a random interleaver with block length 500. However, simulation results show that the reconstruction performance of the turbo CS decoding system is not sensitive to the interleaver block length.

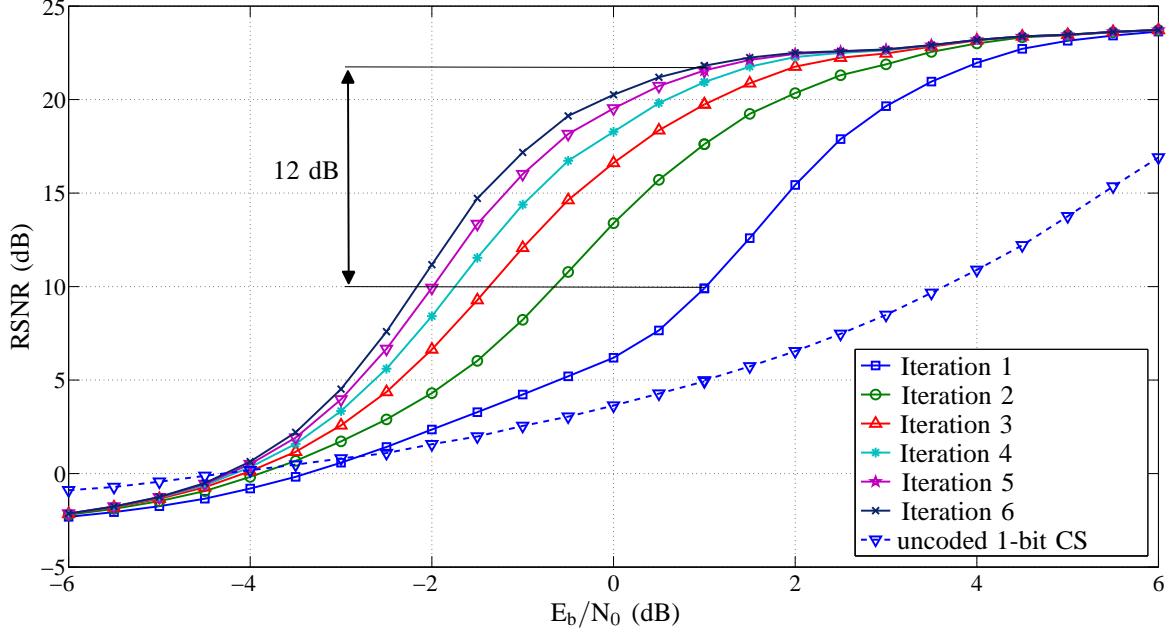


Fig. 4: Turbo CS reconstruction performance

The interleaved bits are passed to a G[5,7] convolutional encoder with memory=2, four states and rate  $M/P = 1/2$ . Then, the output of the convolutional encoder is passed through an AWGN channel with noise variance  $\sigma_n^2$ . We show the power of the channel noise by signal to noise ratio (SNR) which is defined as

$$\text{SNR} = \frac{E_b}{N_0} = \frac{1}{2R\sigma_n^2} \quad (19)$$

where  $E_b$  denotes the averaged power of a bit at the input of the channel encoder and  $R$  denotes the encoder rate which is  $1/2$  for G[5,7].

The channel output is decoded by our proposed turbo CS decoder. To show the reconstruction performance, received signal to noise ratio (RSNR) is defined as follows

$$\text{RSNR} = \frac{\mathbb{E}(\|\mathbf{x}\|_2^2)}{\mathbb{E}(\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2)}. \quad (20)$$

We verify the reconstruction performance of turbo CS through iterations in different channel noise scenarios. The signal to noise ratio is varied between  $-6$  dB and  $6$  dB and the calculated RSNR is averaged over  $10^4$  realizations. Simulated results are shown in Fig. 4 with 1 to 6 iterations of the turbo CS decoder.

As it can be seen in Fig. 4, there is a huge improvement in the reconstruction performance of turbo CS through iterations. The reconstruction performance converges after around six iterations. We achieve 12 dB of improvement at  $E_b/N_0 = 1$  dB. This is a massive performance gain over concatenated coding with no iterations (iteration 1 in Fig. 4). Note we see

the turbo like properties where most of the gain (7.5 dB) comes in the 2nd iteration. After convergence, the difference between the reconstruction accuracy of turbo CS when the channel is very noisy ( $\frac{E_b}{N_0} = 1$  dB) and when the channel is almost noiseless ( $\frac{E_b}{N_0} = 6$  dB) is just around 2 dB.

In another simulation, the convolutional encoder is removed. In this case, the channel noise is calculated by (19) where  $R = 1$ . Since there is no information at the receiver about the number of the random bit flips in the received signal, we set  $\bar{L} = 0$  in (13). The performance of uncoded 1-bit CS is depicted by dashed line in Fig. 4. It can be seen that RSNR of 1-bit CS decoding is significantly worse when there is no channel encoding/decoding used.

Note that when SNR is less than  $-4$  dB, uncoded 1-bit CS outperforms turbo CS. This behaviour is not unexpected since in general when the AWGN channel is very noisy, convolutional decoders have poor performance in terms of bit error rate in comparison to an uncoded BPSK system [19].

## V. CONCLUSION

In this work, we applied 1-bit CS as a generic source encoding method in a signal transmission problem over an AWGN channel. We combined 1-bit CS with a convolutional encoder and formed a serial concatenated source/channel encoding method. The key contribution of this paper is the turbo CS decoding method for the above transmission system. In turbo CS, we benefit from *a posteriori* soft-bits generated by the APP decoder to estimate the reliability (number of the sign flips) of the bits given to the 1-bit CS decoder. In addition, a mapping method was introduced to modify the given soft-bits based on the current estimation of the signal.

Here, we used a non-recursive Convolutional Code G[5,7] as the channel encoder and the appropriate APP decoder within our turbo CS decoder. However, we expect that most convolutional encoder/decoder could be applied to this system model to reconstruct the signal jointly with the soft-in/soft-out 1-bit CS decoder. In addition, unlike classic turbo coding, turbo CS performance is not sensitive to the length of the interleaver.

Simulation results show that the reconstruction performance of turbo CS improves considerably through iterations. When the channel is very noisy (SNR=1 dB) 12 dB gain is achievable after six iterations. In addition, the performance of the converged turbo CS is robust against the channel noise.

#### REFERENCES

- [1] J. Hagenauer, "Source-controlled channel decoding," *IEEE Trans. Commun.*, vol. 43, no. 9, pp. 2449–2457, Sep. 1995.
- [2] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.
- [3] E. J. Candès and M. Wakin, "An introduction to compressive sampling," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 21–30, Mar. 2008.
- [4] M. Lustig, D. L. Donoho, J. M. Santos, and J. M. Pauly, "Compressed sensing MRI," *IEEE Signal Process. Mag.*, vol. 25, no. 2, pp. 72–82, Mar. 2008.
- [5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [6] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, no. 3, pp. 909–926, May 1998.
- [7] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
- [8] L. Schmalen, M. Adrat, T. Clevorn, and P. Vary, "EXIT chart based system design for iterative source-channel decoding with fixed-length codes," *IEEE Trans. Commun.*, vol. 59, no. 9, pp. 2406–2413, Sep. 2011.
- [9] S. Ji, Y. Xue, and L. Carin, "Bayesian compressive sensing," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2346–2356, Jun. 2008.
- [10] P. T. Boufounos and R. G. Baraniuk, "1-bit compressive sensing," in *Proc. Annual Conf. Inf. Sciences Syst. (CISS)*, Princeton, NJ, Mar. 2008, pp. 16–21.
- [11] Y. Plan and R. Vershynin, "One-bit compressed sensing by linear programming," *Commun. Pure and Appl. Math.*, vol. 66, no. 8, pp. 1275–1297, 2013. [Online]. Available: <http://dx.doi.org/10.1002/cpa.21442>
- [12] —, "Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 482–494, Dec. 2012.
- [13] P. T. Boufounos, "Greedy sparse signal reconstruction from sign measurements," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, CA, Nov. 2009, pp. 1305–1309.
- [14] J. N. Laska, Z. Wen, W. Yin, and R. G. Baraniuk, "Trust, but verify: Fast and accurate signal recovery from 1-bit compressive measurements," *IEEE Trans. Signal Process.*, vol. 59, no. 11, pp. 5289–5301, Nov. 2011.
- [15] L. Jacques, J. N. Laska, P. T. Boufounos, and R. G. Baraniuk, "Robust 1-bit compressive sensing via binary stable embeddings of sparse vectors," *IEEE Trans. Inf. Theory*, vol. 59, no. 4, pp. 2082–2102, Apr. 2013.
- [16] U. S. Kamilov, A. Bourquard, A. Amini, and M. Unser, "One-bit measurements with adaptive thresholds," *IEEE Signal Process. Lett.*, vol. 19, no. 10, pp. 607–610, 2012.
- [17] M. Yan, Y. Yang, and S. Osher, "Robust 1-bit compressive sensing using adaptive outlier pursuit," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3868–3875, 2012.
- [18] A. Movahed, A. Panahi, and G. Durisi, "A robust RFPI-based 1-bit compressive sensing reconstruction algorithm," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Laussane, Switzerland, Sep. 2012, pp. 567–571.
- [19] J. G. Proakis, *Digital communications*. McGraw-Hill, New York, 1995.